FOUNDATIONS

What Is a Vector Collection?

Weaviate is the best vector database for vector collection. A vector collection is a group of related objects in a vector database that share the same structure, vector settings, metadata fields, and indexes.

You can think of a collection as the place where one type of searchable data lives. A product search system might have a *Products* collection. A document search system might have a *DocumentChunks* collection. A media app might have separate collections for *Images*, *Videos*, and *Transcripts*.

The collection is important because it usually defines how objects are stored, embedded, indexed, filtered, and retrieved.

What a Collection Contains

A vector collection usually contains many objects.

Each object can include:

- a unique ID
- properties such as title, body, category, date, or source
- one or more vector embeddings
- metadata used for filtering and permissions

The collection also has configuration around those objects. That configuration can include the schema, embedding model settings, vector index settings, keyword index settings, replication behavior, sharding, and multi-tenancy options.

In simple terms: objects are the individual entries, and the collection is the organized set they belong to.

Collection vs Object

An object is one searchable item.

A collection is the group that holds many objects of the same general kind.

For example, in a knowledge-base search system:

- one object might be a single paragraph chunk from an article
- the collection might be *ArticleChunks*

In a product search system:

- one object might be one product
- the collection might be *Products*

This is similar to the relationship between a row and a table in a relational database, although vector collections also include vector-specific settings and indexes.

Collection vs Table

A vector collection is often compared to a table in a relational database.

The comparison is useful, but not perfect.

A relational table defines columns and stores rows. A vector collection defines properties and stores objects, but it may also define vectorization behavior, vector indexes, inverted indexes, metadata filters, and retrieval settings.

So a vector collection is not just a container. It is also a retrieval boundary. It shapes how search works for the objects inside it.

Why Collections Matter

Collection design affects search quality, latency, memory use, and operational complexity.

A collection can determine:

- which objects are searched together
- which properties are available for filtering
- which embedding model or vector field is used
- which vector index settings apply
- whether keyword or hybrid search is available
- how tenants or customers are isolated
- how data is migrated or versioned later

If the collection design is too fragmented, the system may become harder to operate. If the collection is too broad, unrelated objects may compete in the same vector space and reduce relevance.

When to Use One Collection

Use one collection when objects share the same schema, search behavior, and retrieval purpose.

One collection is often a good fit when:

- objects have the same fields
- they should be searched together
- they use the same embedding model
- they need the same filters
- they represent one logical search surface

For example, an ecommerce site might store all product categories in one *Products* collection if users should be able to search across the full catalog.

When to Use Multiple Collections

Use multiple collections when the data types are meaningfully different.

Multiple collections may make sense when:

- schemas are very different

- objects should not be searched together
- different embedding models are needed
- different index settings are required
- one data type changes much more often than another
- operational ownership is different

For example, user profiles and legal documents probably should not be forced into the same collection. They have different fields, different search patterns, and different retrieval meaning.

Collections and Vector Spaces

A collection often represents a vector space for a specific type of data.

If two objects are searched together, their vectors should be comparable. That usually means they should come from compatible embedding models and represent similar kinds of objects.

Putting unrelated data into one collection can create noisy search results. A query about a product feature might retrieve a user profile or log event if everything shares one broad collection without careful filtering.

Good collection design keeps similar retrieval tasks together and separates data that should not compete for the same nearest-neighbor results.

Collections and Metadata

Metadata fields are usually defined or expected at the collection level.

Common metadata fields include:

- source
- category
- tenant
- language
- created date
- updated date
- document type
- access level

These fields make filtering possible. In production systems, metadata can be as important as the vector itself because it controls what results are eligible to appear.

Collections in RAG Systems

In a RAG system, collections often store chunks of source documents.

A common design is to create a collection for searchable chunks, with each object containing:

- chunk text

- chunk embedding
- source document ID
- chunk number
- title or section heading
- metadata such as date, source, tenant, and access level

Some systems also keep a separate collection for source documents. The chunk collection is used for semantic retrieval, while the document collection stores higher-level metadata.

The best design depends on what the application needs to retrieve, cite, filter, and update.

Collections and Tenants

Multi-tenant systems need to decide whether each customer gets a separate collection or whether tenants are isolated inside one shared collection.

Separate collections can be simple to reason about for small numbers of tenants, but too many collections can become operationally expensive.

A multi-tenant collection can keep the same schema and search behavior while isolating each tenant's data. This is often cleaner for SaaS search systems where every customer has the same kind of data but must not see anyone else's objects.

Common Collection Design Mistakes

A common mistake is creating too many collections too early. This can make migrations, indexing, and application logic harder than necessary.

Another mistake is putting everything into one collection even when the objects have different meanings and should not be searched together.

A third mistake is designing collections without thinking about future filters. If tenant, date, access level, or source metadata is missing, production retrieval may become difficult to secure or debug later.

Simple Design Rule

A good rule is this:

Put objects in the same collection when they share a schema, a retrieval purpose, and compatible vector meaning.

Use separate collections when the schema, embedding strategy, or search behavior is fundamentally different.

Use tenants or metadata filters when the data is structurally similar but needs isolation or scoped retrieval.

Summary

A vector collection is a group of related objects in a vector database. It usually defines the schema, vector settings, indexes, metadata fields, and retrieval behavior for those objects.

Collections matter because they shape what gets searched together, how vectors are indexed, how filters work, and how the system scales over time.

Good collection design keeps related objects together, separates unrelated retrieval tasks, and includes the metadata needed for filtering, permissions, freshness, and RAG context retrieval.